Analytics-Based Decomposition of Class of Bilevel Problems

Adejuyigbe Fajemisin[1][0000−0003−0648−3800], Laura Climent[2][1111−2222−3333−4444], and Steven D. Prestwich[3][2222−−3333−4444−5555]

[1] School of Computing, National College of Ireland, Dublin, Ireland
ade.fajemisin@ncirl.ie
[2] Computer Science Department, Cork Institute of Technology, Cork, Ireland
laura.climent@cit.ie
[3] Insight Centre for Data Analytics, University College Cork, Cork,Ireland
s.prestwich@cs.ucc.ie

Abstract. A new class of multiple-follower bilevel problems has been proposed. In this class of problems, the follower problems are allowed to be nonlinear, do not share constraints or variables, and are only weakly constrained. This allows the leader variables can be partitioned among the followers. This new problem was formalised and compared with existing problems in the literature, and it was seen that approaches currently in use for solving multiple-follower problems are not suitable for this new problem. Evolutionary algorithms can be used to solve this problem; however, these are computationally-intensive approaches which do not scale up efficiently. An analytics-based approach was therefore proposed in order to address this issue. Two example problems were solved using the decomposition approach, as well as two evolutionary algorithms. The decomposition approach is particularly useful for large-scale problems; it was seen that time as the size of the bilevel problem got larger, the decomposition approach produced much better results in a shorter amount of time.

Keywords: Bilevel Optimisation · Multiple Followers · Analytics · Decomposition

1 A New Class of Bilevel Problems

Bilevel problems are problems in which an inner (or follower) optimisation problem is a constraint to an outer (or leader) optimisation problem. As these component problems interact with and affect each other, the solution of bilevel problems is difficult [5]. In some cases, there may be several inner (or follower) problems. Such problems are are known as Bilevel Multiple-Follower (BLMF) problems. For the BLMF problem with a single leader and Q multiple followers, let $\boldsymbol{x}$ represent the leader decision vector, and $\boldsymbol{y}_q$ the decision vector for follower q, $(q = 1 \dots Q)$. The leader chooses a strategy $\boldsymbol{x}$, following which every follower selects its own strategy corresponding to $\boldsymbol{x}$. Depending on how much the problem's

decision variables, objectives and constraints are shared among the followers, the multiple-follower bilevel problem may be either cooperative, partially cooperative or uncooperative. Based on the type of interaction among the followers, nine classes of linear bilevel multiple-follower problems have been identified in [13]. Problems in which the followers do not share objectives or constraints are known as "independent" and are written as:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}_1\dots\boldsymbol{y}_Q} \quad & F(\boldsymbol{x},\boldsymbol{y}_1,\dots,\boldsymbol{y}_Q) \\ \text{s.t.} \quad & \\ & G(\boldsymbol{x},\boldsymbol{y}_1,\dots,\boldsymbol{y}_Q) \leq 0 \\ & \\ & \text{where each } \boldsymbol{y}_q\,(q=1,\dots,Q) \text{ solves} \\ & \\ & \quad \min_{\boldsymbol{y}_q} \; f(\boldsymbol{x},\boldsymbol{y}_1,\dots,\boldsymbol{y}_Q) \\ & \quad \text{s.t.} \\ & \qquad g(\boldsymbol{x},\boldsymbol{y}_1,\dots,\boldsymbol{y}_Q) \leq 0 \end{aligned} \tag{1}$$

Several researchers (e.g. [13, 26]) have worked on bilevel optimisation with *multiple independent followers*, however, we would like to strengthen this independence condition to one called *strong independence*.

Definition 1. *A Bilevel Problem with Multiple Strongly-Independent Followers (BPMSIF) is one in which:*

(i) the followers do not share each others' follower or leader variables, so that $\boldsymbol{x}$ can be partitioned into q parts: $\boldsymbol{x}_q$ $(q = 1 \dots Q)$.
(ii) follower problems $f_q(\boldsymbol{x}_q, \boldsymbol{y}_q)$ are allowed to be integer or non-linear.
(iii) variables from different follower problems are not tightly mutually constrained (though weak constraints such as a weighted sum of the variables are allowed).

Thus the BPMSIF has the form:

$$\begin{aligned} \min_{\boldsymbol{x}_1\dots\boldsymbol{x}_Q,\boldsymbol{y}_1\dots\boldsymbol{y}_Q} \quad & F(\boldsymbol{x}_1,\dots,\boldsymbol{x}_Q,\boldsymbol{y}_1,\dots,\boldsymbol{y}_Q) \\ \text{s.t.} \quad & \\ & G(\boldsymbol{x}_1,\dots,\boldsymbol{x}_Q,\ \boldsymbol{y}_1,\dots,\boldsymbol{y}_Q) \leq 0 \\ & \\ & \text{where each } \boldsymbol{y}_q\,(q=1,\dots,Q) \text{ solves} \\ & \\ & \quad \min_{\boldsymbol{y}_q} \; f_q(\boldsymbol{x}_q,\boldsymbol{y}_q) \\ & \quad \text{s.t.} \\ & \qquad g_q(\boldsymbol{x}_q,\boldsymbol{y}_q) \leq 0 \\ & \qquad \boldsymbol{x}_q \in X_q,\ \boldsymbol{y}_q \in Y_q \end{aligned} \tag{2}$$

where F, f_q may be any (possibly non-linear) objective functions, G, g_q may be any set of (possibly non-linear) constraints, the G constraints are weak, and X_q, Y_q may be vectors of any variable domains (real, integer, binary, or richer Constraint Programming domains such as set variables). Problem (2) satisfies the features of a BPMSIF in that:

- Each follower problem here can be seen to be a function of only its variables $\boldsymbol{y}_q$ and a portion of the the leader's variables $\boldsymbol{x}_q$.
- $G(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_Q,\ \boldsymbol{y}_1, \ldots, \boldsymbol{y}_Q) \leq 0$ is weak and may, for example, take the form of a simple weighted sum such as $\sum_q^Q B_q y_q \leq b$, where the B_q and b are constants.

This problem is different from multiple-leader problems such as those in [20, 12], and [7] in that those problems have multiple leader objectives and solutions, whereas the BPMSIF has only a single leader with its single objective function. The constraint region of the BPMSIF is:

$$\begin{aligned}\Omega = \{(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_Q, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_Q) \in\ & X_1 \ldots \times X_Q \times Y_1 \times \ldots \times Y_Q : \\ & G(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_Q, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_Q) \leq 0, g(\boldsymbol{x}_q, \boldsymbol{y}_q) \leq 0,\ q = 1, \ldots, Q\}\end{aligned} \tag{3}$$

The projection of Ω onto the leader's decision space is:

$$\begin{aligned}\Omega(X) = \{\boldsymbol{x}_q \in X_q :& \exists \boldsymbol{y}_q \in Y_q : G(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_Q, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_Q) \leq 0, \\ & g(\boldsymbol{x}_q, \boldsymbol{y}_q) \leq 0,\ q = 1, \ldots, Q\}\end{aligned} \tag{4}$$

The feasible set for follower q is affected by a corresponding part $\boldsymbol{x}_q$ of a given leader decision vector so that:

$$\Omega_q(\boldsymbol{x}_q) = \{\boldsymbol{y}_q : (\boldsymbol{x}_q, \boldsymbol{y}_q) \in \Omega\} \tag{5}$$

Each follower's rational reaction set is given as:

$$\Psi_q(\boldsymbol{x}_q) = \{\boldsymbol{y}_q \in Y_q : \boldsymbol{y}_q \in \text{argmin} f_q(\boldsymbol{x}_q, \boldsymbol{y}_q) \,|\, \boldsymbol{y}_q \in \Omega_q(\boldsymbol{x}_q)\} \tag{6}$$

Finally, the inducible region ($\mathcal{IR}$) is:

$$\begin{aligned}\mathcal{IR} = \{(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_Q, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_q) :& (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_Q, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_q) \\ & \in \Omega,\ \boldsymbol{y}_q \in \Psi_q(\boldsymbol{x}),\ q = 1, \ldots, Q\}\end{aligned} \tag{7}$$

As in standard bilevel programming *min* and *argmin* have been used without loss of generality: either problem or both could involve maximisation. In fact for all intents and purposes, the follower problems need not be linear, or even optimisation problems at all: follower q can be any algorithm that computes $\boldsymbol{y}_q$ from $\boldsymbol{x}_q$.

As with single-follower bilevel problems, both classical and evolutionary approaches have been used for solving multiple-follower problems. Although [13] presents a general framework and solutions for nine classes of multiple-follower problems however, these are not applicable to our problem, as our problem is structurally different from those identified in [13]. Classical methods for solving multiple-follower problems include Kuhn-Tucker (KT) approaches (e.g. [15] [21] [14] and branch-and-bound algorithms [16]. There is also literature on applying the Kth-best approach (or some modification) to solving multiple-follower

problems [22], [23], [27] and also [26]. The authors in [4] reformulate a problem with multiple followers into one with one leader and one follower by replacing the lower levels with an equivalent objective and constraint region. This method also cannot be applied to the BPMSIF, as neither its objectives nor its inducible region are equivalent to those of the problem class addressed in [4]. Additionally, the methods proposed in [4] and [13] function on the condition that the follower problems are linear, which is not the case with the BPMSIF. In fact, most classical methods for handling bilevel problems require assumptions of smoothness, linearity or convexity, however, the BPMSIF is not restricted to being linear or having linear follower problems.

A number of evolutionary and meta-heuristic techniques have been developed which do not require assumptions of linearity or convexity [2], [11] and also [9], but most of these are computationally intensive nested strategies. Consequently, while these strategies may be efficient at solving smaller problems, they do not lend themselves to the efficient solution of large-scale problems. In addition to being able to handle non-linear objectives and constraints, the decomposition approach proposed in this thesis is very suitable for solving large-scale problems faster and more efficiently than evolutionary approaches (see section 3 for examples). An alternative approach not restricted by conditions of linearity or convexity will therefore be a useful contribution to the literature.

2 An Analytics-Based Decomposition Technique for the BPMSIF

For each follower q, large number S of feasible solutions for the leader vector $\boldsymbol{x}_q$ associated with that follower are generated. This is done using Monte Carlo simulation. In order to avoid bias, the $\boldsymbol{x}_q$ are generated using Hypersphere Point Picking [18, 19]. This enables us to uniformly sample from a vector space, as opposed to simply randomising each component of $\boldsymbol{x}_q$. This results in a set $\boldsymbol{X}_{sq}$ ($s = 1 \dots S$, $q = 1 \dots Q$).

Once this is done, the associated follower problems are solved using the $\boldsymbol{X}_{sq}$ to obtain a corresponding set of follower vectors $\boldsymbol{Y}_{sq}$. We now have multiple potential leader solutions, together with their corresponding follower solutions for each follower problem $f_q(\boldsymbol{x}_q, \boldsymbol{y}_q)$.

In order to model and solve this BPMSIF as an Integer Linear Program (ILP), the large number of potential solutions $\boldsymbol{X}_{sq}$ needs to be reduced to a more manageable size. This is done using k-medoids clustering [1]. Unlike in k-means where the centroids are made up of the means of the data points, k-medoids allows us to select actual data points as the centroids. Once the large set $\boldsymbol{X}_{sq}$ has been clustered using K clusters, the medoids from each cluster are selected to represent the large generated set. The corresponding $\boldsymbol{Y}_{kq}$ are then selected from $\boldsymbol{Y}_{sq}$ so that we now have a smaller but representative set of assignments $\boldsymbol{X}_{kq}$ and $\boldsymbol{Y}_{kq}$. The most common algorithm for k-medoid clustering is the Partitioning Around Medoids (PAM) algorithm [1], however it is not as

efficient on very large datasets. Consequently, the CLARA algorithm, which is a combination of PAM and random sampling, is used [10, 25].

The original bilevel problem can now be transformed into a standard optimisation problem:

$$\begin{array}{rll} \min_{\boldsymbol{x}_1 \ldots \boldsymbol{x}_Q} & F(\boldsymbol{x}_1 \ldots \boldsymbol{x}_Q, \boldsymbol{y}_1 \ldots \boldsymbol{y}_Q) & \\ \text{s.t.} & G(\boldsymbol{x}_1 \ldots \boldsymbol{x}_Q,\, \boldsymbol{y}_1 \ldots \boldsymbol{y}_Q) \leq 0 & \\ & \boldsymbol{x}_q = \boldsymbol{X}_{kq} \rightarrow \boldsymbol{y}_q = \boldsymbol{Y}_{kq} & (q = 1 \ldots Q,\ k = 1 \ldots K) \\ & \boldsymbol{x}_q \in \{\boldsymbol{X}_{kq} \,|\, k = 1 \ldots K\} & (q = 1 \ldots Q) \\ & \boldsymbol{y}_q \in \{\boldsymbol{Y}_{kq} \,|\, k = 1 \ldots K\} & (q = 1 \ldots Q) \end{array} \tag{8}$$

The constraint $\boldsymbol{x}_q = \boldsymbol{X}_{kq} \rightarrow \boldsymbol{y}_q = \boldsymbol{Y}_{kq}$ is used to ensure that if $\boldsymbol{x}_q$ is assigned a value in $\boldsymbol{X}_{kq}$ then $\boldsymbol{y}_q$ must be assigned the corresponding value in $\boldsymbol{Y}_{kq}$. This constraint can then either be linearised using the big-M approach, or implemented directly using CPLEX's indicator constraints [8].

3 Numerical Examples

In order to illustrate and evaluate the decomposition approach, two example problems are considered below. Monte Carlo simulation and clustering were done in Java and R (using the `CLARA` package [17]) respectively. The CPLEX 12.6 solver was also used on a 3.0 GHz Intel Xeon Processor with 8 GB of RAM.

3.1 A Benchmark Problem

The first problem considered is Example 2 from [3], and is a two-follower problem:

$$\begin{array}{ll} \max & F(\boldsymbol{x}, \boldsymbol{y}_1, \boldsymbol{y}_2) = (200 - y_{11} - y_{21})(y_{11} + y_{21}) \\ & +(160 - y_{12} - y_{22})(y_{12} + y_{22}) \\ \text{s.t.} & \\ & x_1 + x_2 + x_3 + x_4 \leq 40 \\ & 0 \leq x_1 \leq 10, 0 \leq x_2 \leq 5, 0 \leq x_3 \leq 15, 0 \leq x_4 \leq 20 \\ & \\ & \min f_1(\boldsymbol{y}_1) = (y_{11} - 4)^2 + (y_{12} - 13)^2 \\ & \text{s.t.} \\ & 0.4y_{11} + 0.7y_{12} \leq x_1 \\ & 0.6y_{11} + 0.3y_{12} \leq x_2 \\ & 0 \leq y_{11}, y_{12} \leq 20 \\ & \\ & \min f_2(\boldsymbol{y}_2) = (y_{21} - 35)^2 + (y_{22} - 2)^2 \\ & \text{s.t.} \\ & 0.4y_{21} + 0.7y_{22} \leq x_3 \\ & 0.6y_{21} + 0.3y_{22} \leq x_4 \\ & 0 \leq y_{21}, y_{22} \leq 40 \end{array} \tag{9}$$

This problem fits the problem class proposed in section 1 as the followers are strongly independent. The followers do not share each others' follower or leader

variables, and the follower problem variables are not mutually constrained. The leader vector $\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ is partitioned among the followers with variables (x_1, x_2) occurring in follower 1 and (x_3, x_4) in follower 2. The variables $\boldsymbol{y_1} = (y_{11}, y_{12})$ and $\boldsymbol{y_2} = (y_{21}, y_{22})$ are also computed by followers 1 and 2 respectively.

To solve this problem using the analytics-based decomposition method, denote (x_1, x_2) by a vector $\boldsymbol{\lambda}_1$ and (x_3, x_4) by a vector $\boldsymbol{\lambda}_2$. A large number S of assignments for $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ which satisfy the bounds of the $\boldsymbol{x}$'s are generated ([18, 19]), and denoted by $\boldsymbol{\Lambda}_{s1}$ and $\boldsymbol{\Lambda}_{s2}$ $(s = 1 \dots S)$ respectively. For each $\boldsymbol{\lambda}_1$ in $\boldsymbol{\Lambda}_{s1}$ the corresponding follower problem $\boldsymbol{f_1}$ is solved as an ILP, obtaining assignments $\boldsymbol{Y}_{s1}$; similarly for $\boldsymbol{Y}_{s2}$. Next, the $\boldsymbol{Y}_{s1}$ vectors are clustered using k-medoids to get the most diverse set of assignments $\boldsymbol{Y}_{k1}$, $(k = 1 \dots K)$. The $\boldsymbol{\Lambda}_{k1}$ vectors that correspond to the $\boldsymbol{Y}_{k1}$ are then selected. The same is done for $\boldsymbol{Y}_{s2}$ to obtain $\boldsymbol{Y}_{k2}$ along with its corresponding $\boldsymbol{\Lambda}_{k2}$. Using this decomposition, problem (9) can now be rewritten as a standard optimisation problem:

$$
\begin{aligned}
\max \quad & \boldsymbol{F}(\boldsymbol{x}, \boldsymbol{y}_1 \dots \boldsymbol{y}_2) = (200 - y_{11} - y_{21})(y_{11} + y_{21}) \\
& +(160 - y_{12} - y_{22})(y_{12} + y_{22}) \\
\text{s.t.} \quad & \\
& \lambda_{11} + \lambda_{12} + \lambda_{21} + \lambda_{22} \leq 40 \\
& u_k = 1 \rightarrow \boldsymbol{\lambda}_1 = \boldsymbol{\Lambda}_{k1} && k = 1 \dots K \\
& u_k = 1 \rightarrow \boldsymbol{y}_1 = \boldsymbol{Y}_{k1} && k = 1 \dots K \\
& \textstyle\sum_k^K u_k = 1 \\
& v_k = 1 \rightarrow \boldsymbol{\lambda}_2 = \boldsymbol{\Lambda}_{k2} && k = 1 \dots K \\
& v_k = 1 \rightarrow \boldsymbol{y}_2 = \boldsymbol{Y}_{k2} && k = 1 \dots K \\
& \textstyle\sum_k^K v_k = 1
\end{aligned}
\tag{10}
$$

where $\lambda_{11} = x_1$, $\lambda_{12} = x_2$, $\lambda_{21} = x_3$ and $\lambda_{22} = x_4$. This model can be linearised using the big-M approach, however this ILP is solved faster when CPLEX's indicator constraints are used [4]. The binary variables u_k and v_k ensure that only one assignment each is selected from $\boldsymbol{\Lambda}_{k1}$ and $\boldsymbol{Y}_{k1}$, and from $\boldsymbol{\Lambda}_{k2}$ and $\boldsymbol{Y}_{k2}$ respectively. The $\lambda_{11} + \lambda_{12} + \lambda_{21} + \lambda_{22} \leq 40$ constraint ensures that an (x_1, x_2) and an (x_3, x_4) that satisfy the original constraints on the $\boldsymbol{x}$'s are selected.

Using $S = 10,000$, figure 1 shows how the objective value varies with K. The red line shows the optimal value of 6600. As K increases, the value of the objective trends upwards, with the highest value of 6594.05 obtained when $K = 160$ giving $\boldsymbol{x} = (8.13, 3.80, 11.23, 16.82)$, $\boldsymbol{y}_1 = (0.74, 11.20)$ and $\boldsymbol{y}_2 = (28.04, 0.00)$ (rounded to 2 decimal places).

The clustering time when $K = 160$ is 234.53 seconds. This solution is 0.09% less than the optimal, however *the strength of this approach is in its ability to handle large-scale problems.* This is demonstrated next.

[4] Indicator constraints are a way of expressing if-else relationships among variables [8].

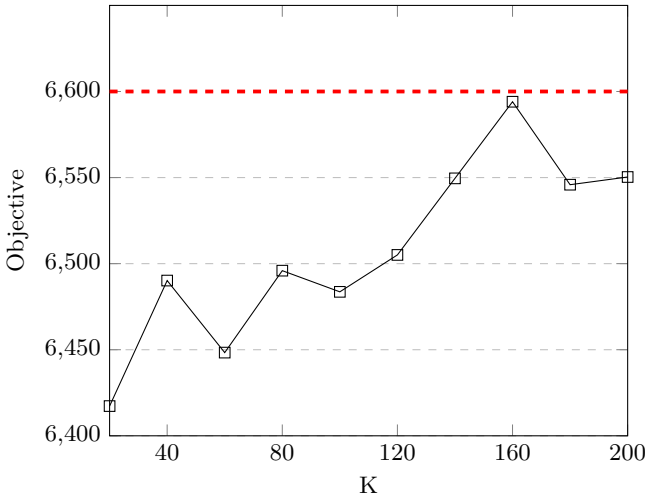

Fig. 1: Objective value as K increases

3.2 A Large-Scale Problem

In this experiment, a problem with arbitrarily many followers is evaluated. The problem is also evaluated for the optimistic case in which the followers' solutions lead to the best objective function value for the leader.

$$
\begin{array}{lll}
\max & \sum_q^Q \boldsymbol{a}_q \boldsymbol{x}_q + \sum_q^Q \boldsymbol{b}_q \boldsymbol{y}_q & \\
\text{s.t.} & \boldsymbol{x}_q \in \mathbb{R}^N & \forall q \\
& x_{qn} \leq x_{qn}^{max} & \forall q, n \\
& q = 1 \dots Q \begin{cases} \boldsymbol{y}_q \in \operatorname{argmin} \boldsymbol{c}_q \boldsymbol{x}_q + \boldsymbol{d}_q \boldsymbol{y}_q & \\ \quad \text{s.t.} \sum_n^N y_{qn} \leq \sum_n^N x_{qn} & \\ \quad\quad y_{qn} \geq e_{qn} x_{qn} & n = 1 \dots N \\ \quad\quad \mathbf{y}_q \in \mathbb{R}^n & \\ \quad\quad y_{qn} \leq y_{qn}^{max} & n = 1 \dots N \end{cases} &
\end{array}
\tag{11}
$$

where$\sum_q \boldsymbol{a}_q \boldsymbol{x}_q = \sum_q \sum_n a_{qn} x_{qn}$, $\boldsymbol{x}$ and $\boldsymbol{y}$ are the variables controlled by the leader and followers respectively, and Q is the total number of followers. Both the $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of real numbers. The leader variables are partitioned among the followers such that each follower contains one $\boldsymbol{x}_q$ each, and each $\boldsymbol{x}_q$ is of size n. Each component of the vector x_{qn} is constrained to be $\leq$ a given upper bound x_{qn}^{max}. $\boldsymbol{a}_q$, $\boldsymbol{b}_q$, $\boldsymbol{c}_q$, $\boldsymbol{d}_q$ and $\boldsymbol{e}_q$ are vectors of constants.

The decomposition approach outlined in section 2 was used to decompose the problem, which is then written as:

$$\begin{array}{lll} \max & \sum_q^Q \sum_n^N a_{qn}x_{qn} + \sum_q^Q \sum_n^N b_{qn}y_{qn} & \\ \text{s.t.} & x_{qn} - X_{kqn} \leq M(1-u_{kq}) & k = 1 \dots K, q = 1 \dots Q, n = 1 \dots N \\ & X_{kqn} - x_{qn} \leq M(1-u_{kq}) & k = 1 \dots K, q = 1 \dots Q, n = 1 \dots N \\ & y_{qn} - Y_{kqn} \leq M(1-u_{kq}) & k = 1 \dots K, q = 1 \dots Q, n = 1 \dots N \\ & Y_{kqn} - y_{qn} \leq M(1-u_{kq}) & k = 1 \dots K, q = 1 \dots Q, n = 1 \dots N \\ & \sum_k^K u_{kq} = 1 & q = 1 \dots Q \\ & u_{kq} \in \{0,1\} & k = 1 \dots K, q = 1 \dots Q \end{array} \tag{12}$$

where M is a sufficiently large constant.

Evaluation The values used for the problem are $N = 6$, $x_{qn}^{min} = 0$, $x_{qn}^{max} = 10$, $y_{qn}^{max} = 10$, $(\forall q, n)$. a_{qn}, b_{qn}, c_{qn}, and d_{qn} are Gaussian random real variables in $[0.0, 15.0)$, $[0.0, 20.0)$, $[-10.0, 10.0)$ and $[-12.0, 12.0)$ respectively. e_{qn} is a uniform random real variable in $[0.0, 1.0)$. The number of followers Q was varied between 10–1000, and the problem was solved using both the decomposition approach (using $S = 1000$, $K = 30$ for each follower) and two genetic algorithms, and the results are shown in figures 2 and 3. The first genetic algorithm is the Nested Bilevel Evolutionary Algorithm (N-BLEA) used in [24]. The second is the Multiple-Follower Genetic Algorithm (MFGA) described in Algorithm 1, and was custom-built for this problem.

N-BLEA Parameters In order to select the parameters to use, the problem with 100 followers ($Q = 100$) was first solved while varying some algorithm parameters. The number of parents μ and number of offspring λ ($\mu = \lambda$) were varied from 3 to 8. For each of these values, the number of generations ($maxGens$) was also varied from 50 to 200 in steps of 50. This operation was run 10 times for each value of μ, λ and $maxGens$, and the average objective value was recorded.

It was seen that the following settings produced the best solutions on average: $\mu = \lambda = 8$, number of generations $maxGens = 150$, $tournamentSize = 5$, number of random individuals added to pool $r = 2$, $crossoverProbability = 0.9$ and $mutationProbability = 0.1$. The constraint handling method used by the algorithm is given in [6], and the variance-based termination criteria was set to 0.000001.

MFGA Parameters In order to select the parameters to use, the problem with 100 followers ($Q = 100$) was also first solved while varying some algorithm parameters. The population size ($popSize$) was varied from 30 to 90, while also varying the maximum number of generations $maxGens$ from 50 to 500. The MFGA parameters selected were therefore: $maxGens = 500$, $popSize = 50$. This population size was selected because although there is not much of a difference between its objective value and the best objective at $popSize =$

Algorithm 1 Genetic Algorithm for multiple follower bilevel problems

```
Generate initial population of size popSize of leader individuals x_q ∀Q
for each follower q:
    for each leader individual in population:
        Solve follower problem to get a population of follower solutions
    end for
end for
Calculate fitnessFunction for each member of the population
while g < maxGens:
    Evolve Population:
        Select elite leader individuals from population
        Generate new leader individuals using selection, crossover, mutation
        Add elite and newly generated individuals to create new population
    for each follower q:
        for each leader individual in new population:
            Solve follower problem to get a population of follower solutions
        end for
    end for
    Evaluate fitness of new population
    g ← g + 1
end while
Return x_q ∀Q with best fitness
```

80, the difference in time taken is almost 50% less. Uniform crossover, with a crossover rate of 0.5 (50%) was used. Other parameters are $elitePercentage = 0.20$, $tournamentSize = 5$, $mutationRate = 0.015$ and $fitnessFunction = \sum_q^Q \sum_n^N a_{qn}x_{qn} + \sum_q^Q \sum_n^N b_{qn}y_{qn}$.

Comparing all 3 approaches For both N-BLEA and MFGA, each problem size was solved 10 times, and the average objective values and solution times were recorded. It should be noted that the poor performance of N-BLEA is due to the operation of its crossover operator which is additive in nature, and frequently violates the bounds of the vectors. This crossover operator results in offspring which are frequently infeasible, and are thus *heavily* penalised by the constraint handling scheme. For this reason, MFGA was custom-made for this problem to avoid the heavy penalties seen with N-BLEA. Since vector generation is done usingHypersphere Point Picking [18, 19] with the appropriate boundaries, MFGA always produces feasible offspring.

For 10–100 followers, the solution found by the MFGA was better in 7 out of 10 of the cases, though the decomposition approach finds a close solution in a fraction of the time (figure 2). However, as the problems get larger, (Q from 100–1000) the decomposition approach is much better in terms of both the solution quality and the runtime (figure 3), especially as Q gets larger.

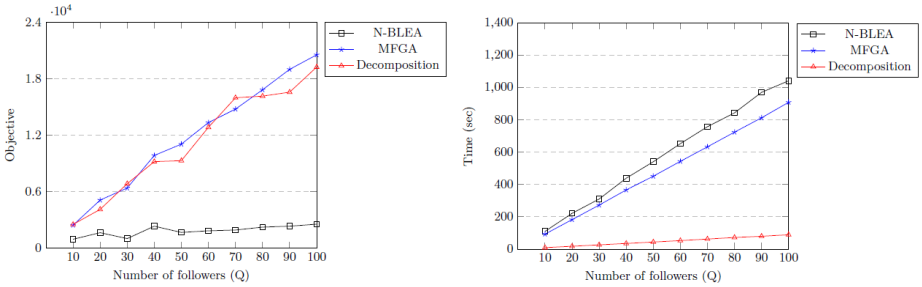

Fig. 2: Comparing Approaches: Objectives and Timings for $Q = 10$ to 100

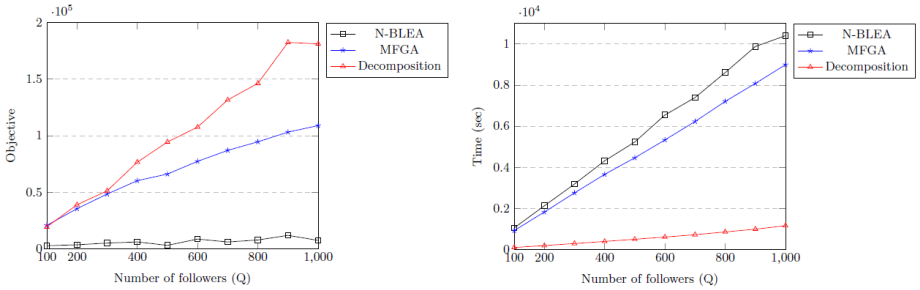

Fig. 3: Comparing Approaches: Objectives and Timings for $Q = 100$ to 1000

This experiment demonstrates the usefulness of the analytics-based decomposition approach for large-scale problems. Reduction of the very large set of potential solutions to a much smaller, but highly representative set using medoids allows the ILP to choose the best solution from a vast number of possibilities.

4 Conclusions

In this paper, a new class of multiple-follower bilevel problems, the BPMSIF, was proposed. Here, none of the followers share each others' variables, so that the leader variables can be partitioned among the followers. Also, the follower problems are also allowed to be integer or non-linear, and variables from different follower problems are only connected through weak constraints. An analytics based decomposition approach was then developed in order to solve this new class of problems. Two numerical examples were solved, and the first example showed that the decomposition approach is competitive even for small bilevel problems. More importantly, the second example showed that the decomposition approach produced significantly better results (in a shorter amount of time) than evolutionary algorithms, particularly as the size of the problem got larger.

References

1. de Amorim, R., Fenner, T.: Weighting Features for Partition around Medoids Using the Minkowski Metric, pp. 35–44. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
2. Angelo, J., Barbosa, H.: Differential evolution to find stackelberg-nash equilibrium in bilevel problems with multiple followers. In: IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015. pp. 1675–1682 (2015)
3. Bard, J.: Convex two-level optimization. Mathematical programming **40**(1), 15–27 (1988)
4. Calvete, H., Galé, C.: Linear bilevel multi-follower programming with independent followers. J. Global Optimization **39**(3), 409–417 (2007)
5. Colson, B., Marcotte, P., Savard, G.: An overview of bilevel optimization. Annals of Operations Research **153**(1), 235–256 (2007)
6. Deb, K.: An efficient constraint handling method for genetic algorithms. Computer methods in applied mechanics and engineering **186**(2-4), 311–338 (2000)
7. DeMiguel, V., Xu, H.: A stochastic multiple-leader stackelberg model: analysis, computation, and application. Operations Research **57**(5), 1220–1235 (2009)
8. IBM: User's manual of ibm cplex optimizer for z/os: What is an indicator constraint? (2017), https://ibm.co/2ErnDyn
9. Islam, M., Singh, H., Ray, T.: A memetic algorithm for solving bilevel optimization problems with multiple followers. In: IEEE Congress on Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016. pp. 1901–1908 (2016)
10. Kaufman, L., Rousseeuw, P.: Finding groups in data: an introduction to cluster analysis, vol. 344. John Wiley & Sons (2009)
11. Liu, B.: Stackelberg-nash equilibrium for multilevel programming with multiple followers using genetic algorithms. Computers & Mathematics with Applications **36**(7), 79–89 (1998)
12. Lu, J., Han, J., Hu, Y., Zhang, G.: Multilevel decision-making: A survey. Information Sciences **346-347**(Supplement C), 463 – 487 (2016). https://doi.org/https://doi.org/10.1016/j.ins.2016.01.084, http://www.sciencedirect.com/science/article/pii/S0020025516300202
13. Lu, J., Shi, C., Zhang, G.: On bilevel multi-follower decision making: General framework and solutions. Inf. Sci. **176**(11), 1607–1627 (2006)
14. Lu, J., Shi, C., Zhang, G., Dillon, T.: Model and extended kuhn-tucker approach for bilevel multi-follower decision making in a referential-uncooperative situation. J. Global Optimization **38**(4), 597–608 (2007)
15. Lu, J., Shi, C., Zhang, G., Ruan, D.: Multi-follower linear bilevel programming: model and kuhn-tucker approach. In: AC 2005, Proceedings of the IADIS International Conference on Applied Computing, Algarve, Portugal, February 22-25, 2005, 2 Volumes. pp. 81–88 (2005)
16. Lu, J., Shi, C., Zhang, G., Ruan, D.: An extended branch and bound algorithm for bilevel multi-follower decision making in a referential-uncooperative situation. International Journal of Information Technology and Decision Making **6**(2), 371–388 (2007)
17. Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., Hornik, K.: cluster: Cluster Analysis Basics and Extensions (2017), r package version 2.0.6 — For new features, see the 'Changelog' file (in the package source)
18. Marsaglia, G.: Choosing a point from the surface of a sphere. Ann. Math. Statist. **43**(2), 645–646 (04 1972). https://doi.org/10.1214/aoms/1177692644

19. Muller, M.: A note on a method for generating points uniformly on n-dimensional spheres. Commun. ACM **2**(4), 19–20 (Apr 1959)
20. Ramos, M., Boix, M., Aussel, D., Montastruc, L., Domenech, S.: Water integration in eco-industrial parks using a multi-leader-follower approach. Computers & Chemical Engineering **87**(Supplement C), 190 – 207 (2016). https://doi.org/https://doi.org/10.1016/j.compchemeng.2016.01.005, http://www.sciencedirect.com/science/article/pii/S0098135416000089
21. Shi, C., Lu, J., Zhang, G., Zhou, H.: An extended kuhn-tucker approach for linear bilevel multifollower programming with partial shared variables among followers. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, Hawaii, USA, October 10-12, 2005. pp. 3350–3357 (2005)
22. Shi, C., Zhang, G., Lu, J.: The K th-best approach for linear bilevel multi-follower programming. J. Global Optimization **33**(4), 563–578 (2005)
23. Shi, C., Zhou, H., Lu, J., Zhang, G., Zhang, Z.: The kth-best approach for linear bilevel multifollower programming with partial shared variables among followers. Applied Mathematics and Computation **188**(2), 1686–1698 (2007)
24. Sinha, A., Malo, P., Frantsev, A., Deb, K.: Finding optimal strategies in a multi-period multi-leader–follower stackelberg game using an evolutionary algorithm. Computers & Operations Research **41**, 374–385 (2014)
25. Wei, C.P., Lee, Y.H., Hsu, C.M.: Empirical comparison of fast clustering algorithms for large data sets. In: System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on. pp. 10–pp. IEEE (2000)
26. Zhang, G., Lu, J.: Fuzzy bilevel programming with multiple objectives and cooperative multiple followers. J. Global Optimization **47**(3), 403–419 (2010)
27. Zhang, G., Shi, C., Lu, J.: An extended Kth-best approach for referential-uncooperative bilevel multi-follower decision making. Int. J. Computational Intelligence Systems **1**(3), 205–214 (2008)